Class 07: Machine Learning

Lana (PID:A17013518)

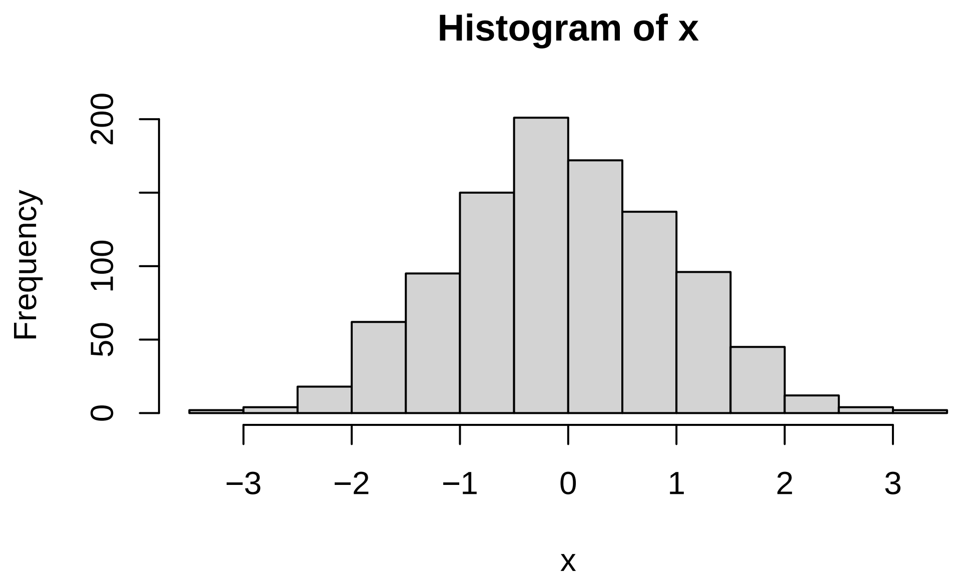
Clustering Methods

The broad goal here is to find grouping (clusters) in your input data.

Kmeans

First, let's make up some data to cluster.

```
x <- rnorm(1000)
hist(x)
```



Make a vector of length 60 with 30 points centered at -3 and 30 points centered at +3

```r
tmp <- c(rnorm(30, mean = -3), rnorm(30, mean=3))
tmp
```

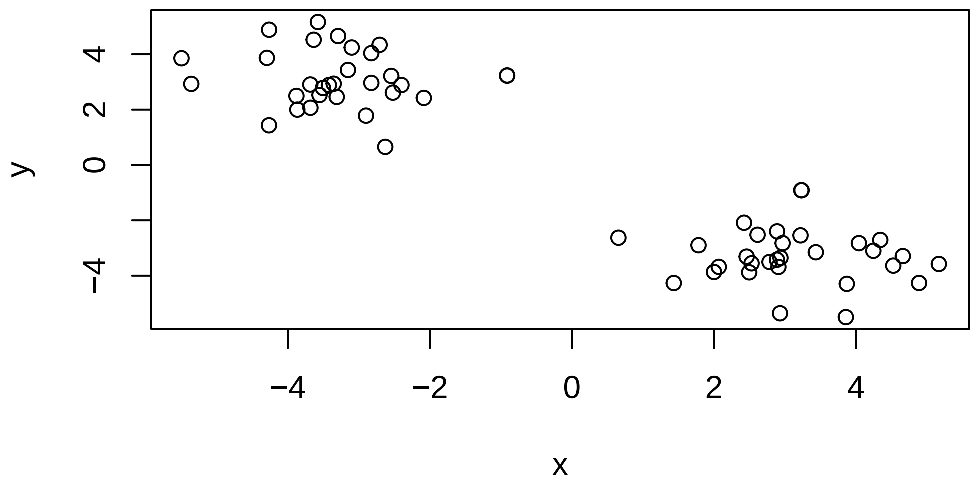
```
 [1] -4.2636379 -4.2655327 -3.2921880 -3.1532843 -2.0853496 -3.3542151
 [7] -2.6278981 -3.5762702 -3.8662051 -3.5534463 -2.7073550 -4.2948711
[13] -2.5434864 -3.8787771 -2.8245481 -3.1000712 -2.3996220 -5.4969447
[19] -3.6359823 -3.4217302 -2.5206616 -3.6814908 -3.3113487 -3.6836124
[25] -0.9114059 -3.5052324 -5.3590019 -2.8990608 -0.9126699 -2.8240840
[31]  2.9669712  3.2297234  1.7828317  2.9304934  2.7824536  3.2346053
[37]  2.9082036  2.4601156  2.0678986  2.6143794  2.8871483  4.5250711
[43]  3.8571460  2.8892992  4.2444765  4.0408944  2.4964628  3.2191740
[49]  3.8705715  4.3421666  2.5293236  2.0007628  5.1667340  0.6551775
[55]  2.9369875  2.4235691  3.4345135  4.6592965  1.4338930  4.8880406
```

I will now make a wee x and y dataset with 2 groups of points.

```r
rev(c(1:5))
```

```
[1] 5 4 3 2 1
```

```r
x <- cbind(x=tmp, y=rev(tmp))
plot(x)
```



```
k <- kmeans(x, centers = 2)
k
```

```
K-means clustering with 2 clusters of sizes 30, 30

Cluster means:
          x         y
1  3.115946 -3.264999
2 -3.264999  3.115946

Clustering vector:
 [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1
[39] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

Within cluster sum of squares by cluster:
[1] 61.33383 61.33383
 (between_SS / total_SS =  90.9 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
```

Q. From your result object `k` how many ponts are in each cluster?

```
k$size
```

```
[1] 30 30
```

Q. What "component" of your result object details the cluster membership?

```
k$cluster
```

```
 [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1
[39] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

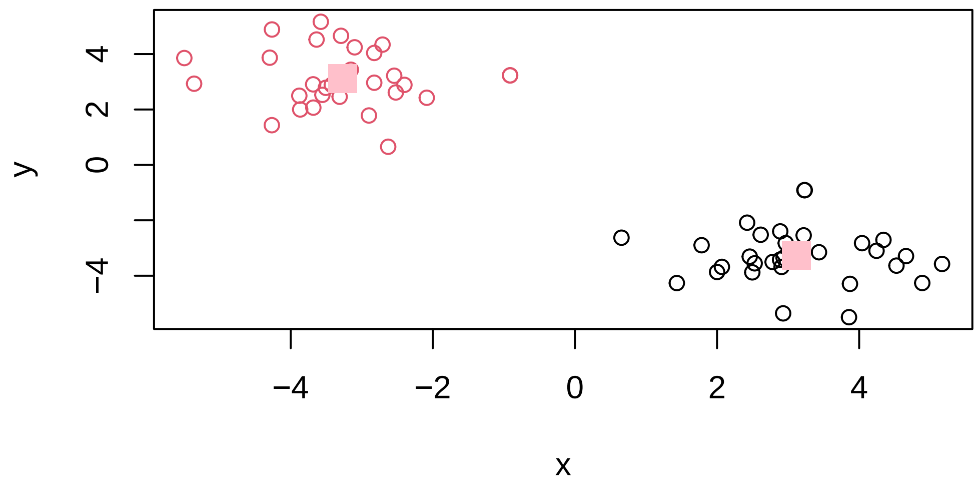
Q. Cluster centers?

```
k$centers
```

```
          x         y
1  3.115946 -3.264999
2 -3.264999  3.115946
```

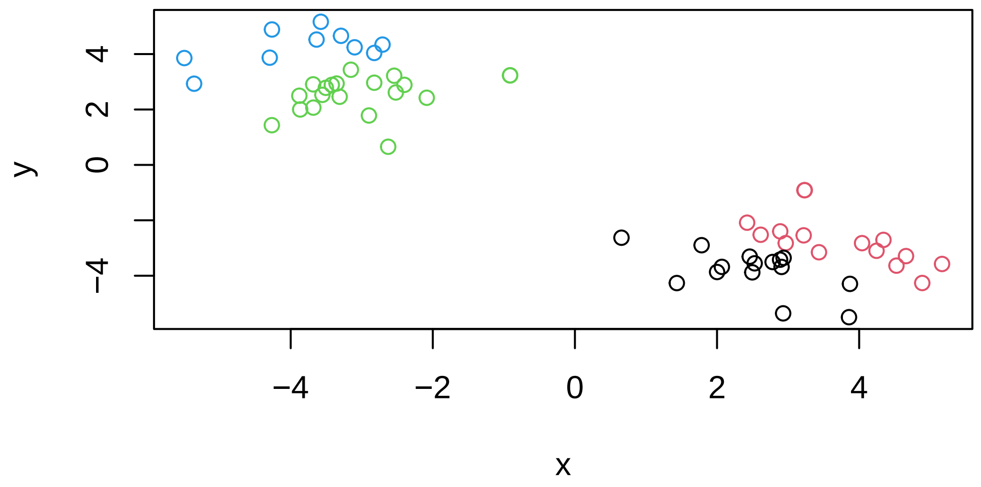
Plot for clustering results

```
plot(x, col = k$cluster)
points(k$centers, col = "pink", pch = 15, cex = 2)
```



We can cluster into 4 groups

```
# kmeans
k4 <- kmeans(x, centers = 4)
# plot results
plot(x, col = k4$cluster)
```



A big limitation of kmeans is that it does what you ask even if you ask for silly clusters.

Hierarchical Clustering

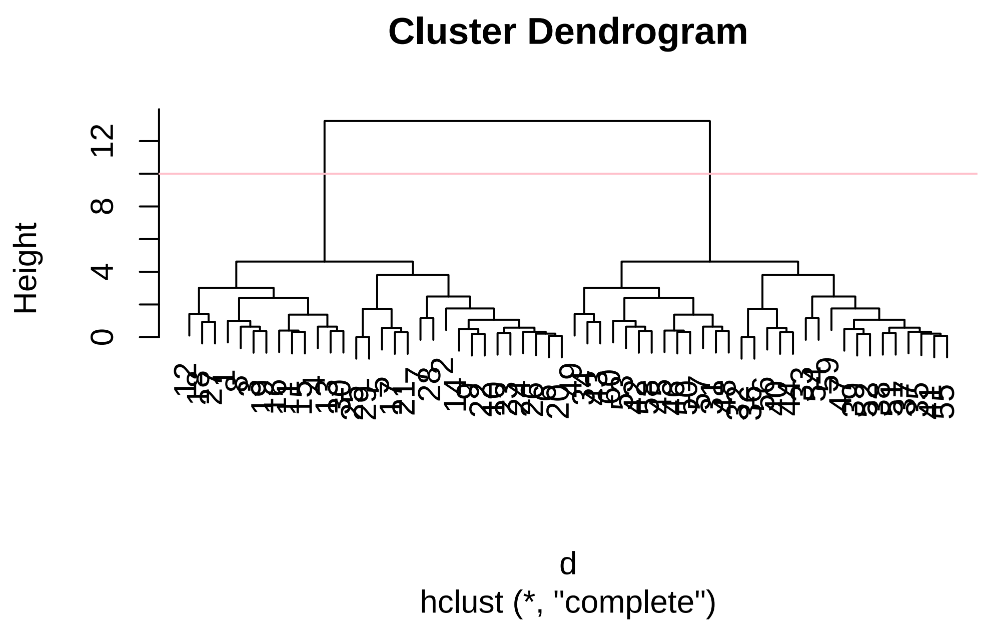
The main base R function for Hierarchical Clustering is `hclust()`. Unlike `kmeans()` you can not just pass it your data as input. You first need to calculate distance matrix.

```r
d <- dist(x)
hc <- hclust(d)
hc
```

```

Call:
hclust(d = d)

Cluster method   : complete 
Distance         : euclidean 
Number of objects: 60 
```

Use `plot()` to view results

```r
plot(hc)
abline(h=10, col = "pink")
```



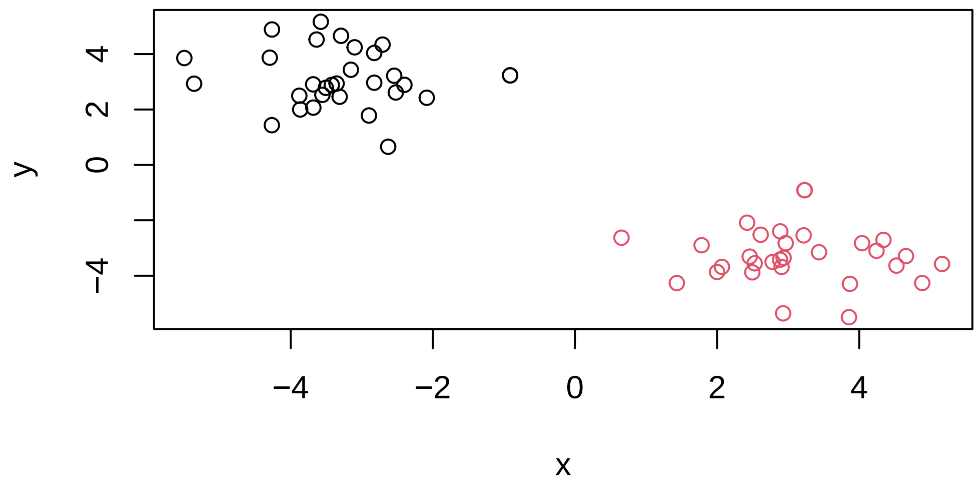
To make the "cut" and get our cluster membership vector we can use the `cutree()` function.

```r
grps <- cutree(hc, h=10)
grps
```

```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

Make a plot of data colored by hclust results

```r
plot(x, col=grps)
```



principal Component Analysis (PCA)

Here we will do principal component analysis (PCA) on some food data from the UK.

```r
url <- "https://tinyurl.com/UK-foods"
x <- read.csv(url, row.names = 1)
```

```r
# rownames(x) <- x[,1]
# x <- x[, -1]
# x
```

> Q1. How many rows and columns are in your new data frame named x? What R functions could you use to answer this questions?

```r
dim(x)
```

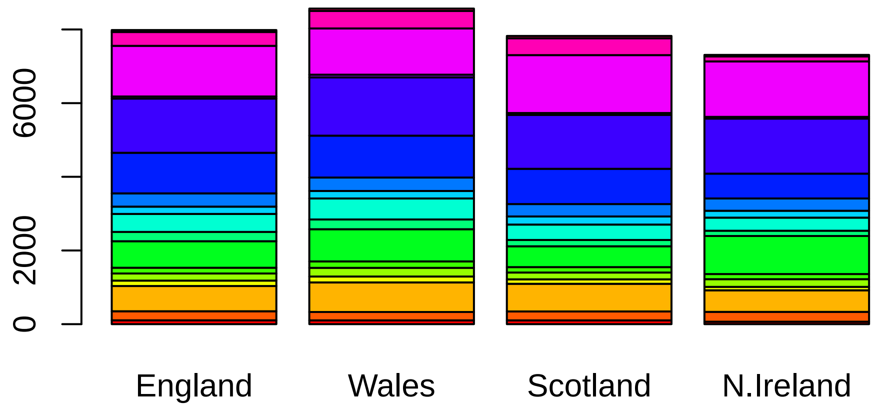
```
[1] 17  4
```

> Q2. Which approach to solving the 'row-names problem' mentioned above do you prefer and why? Is one approach more robust than another under certain circumstances?

```
# I prefer the first approach just because it looks more simple and clean in the code and
```

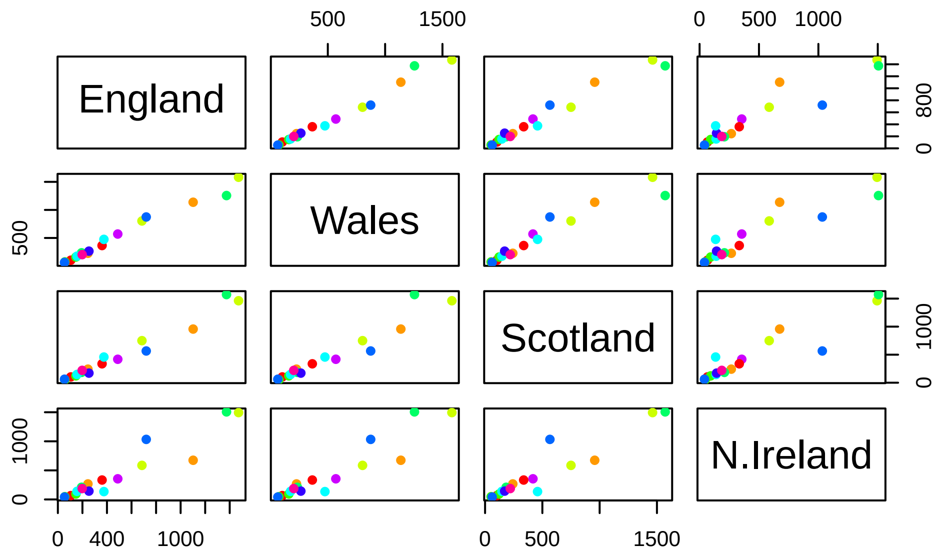
Q3: Changing what optional argument in the above barplot() function results in the following plot?

```
# Changing beside to false results in changing the barplot because if false, columns of he
barplot(as.matrix(x), beside=F, col=rainbow(nrow(x)))
```



Q5: Generating all pairwise plots may help somewhat. Can you make sense of the following code and resulting figure? What does it mean if a given point lies on the diagonal for a given plot?

```
pairs(x, col=rainbow(10), pch=16)
```



Q6. What is the main differences between N. Ireland and the other countries of the UK in terms of this data-set?

PCA to the Rescue

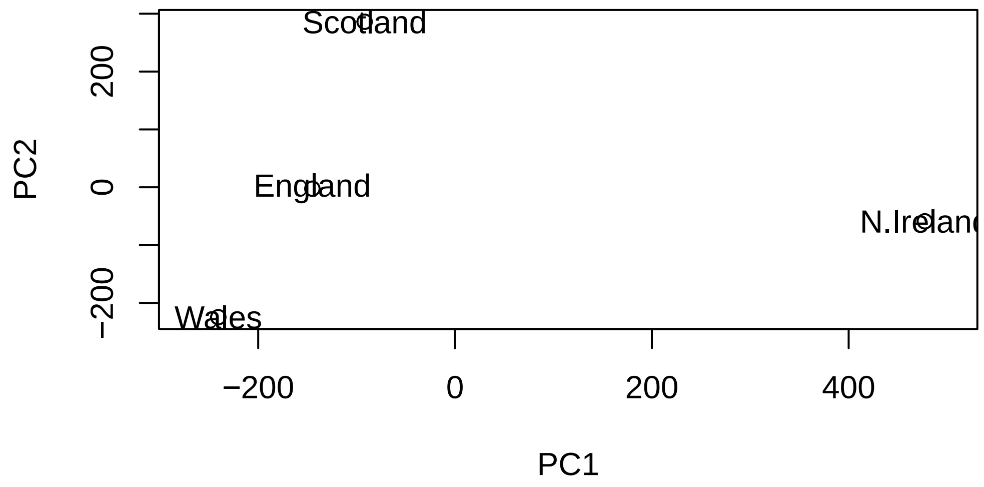
The main "base" R function of PCA is called `prcomp()`.

```r
pca <- prcomp(t(x))
summary(pca)
```

```
Importance of components:
                            PC1      PC2      PC3       PC4
Standard deviation     324.1502 212.7478 73.87622 2.921e-14
Proportion of Variance   0.6744   0.2905  0.03503 0.000e+00
Cumulative Proportion    0.6744   0.9650  1.00000 1.000e+00
```

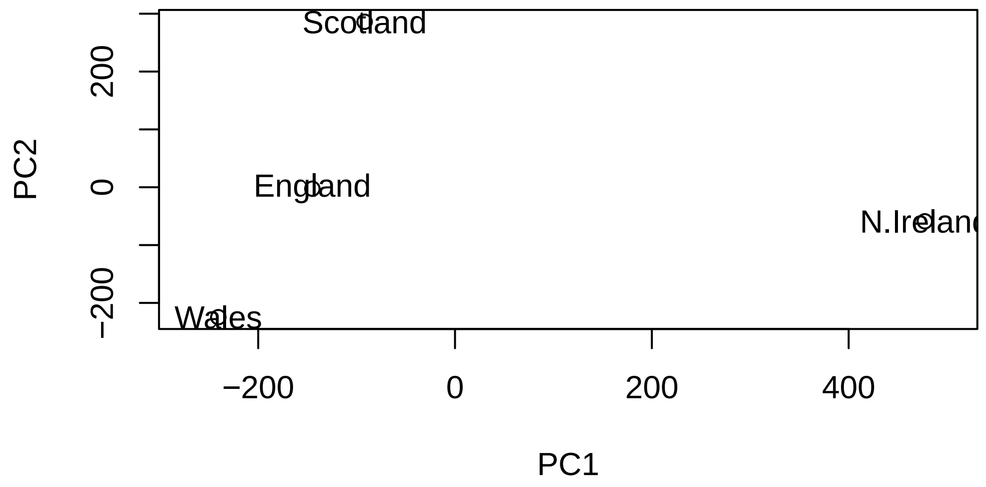
Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

```r
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2", xlim=c(-270,500))
text(pca$x[,1], pca$x[,2], colnames(x))
```



Q8. Customize your plot so that the colors of the country names match the colors in our UK and Ireland map and table at start of this document.

```
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2", xlim=c(-270,500))
text(pca$x[,1], pca$x[,2], colnames(x))
```



Q. How much variance is captured in 2 PCs

96.5%

To make our main “PC score plot” or “PC1 vs PC2 plot”, or “PC plot” or “ordination plot”

```
attributes(pca)
```

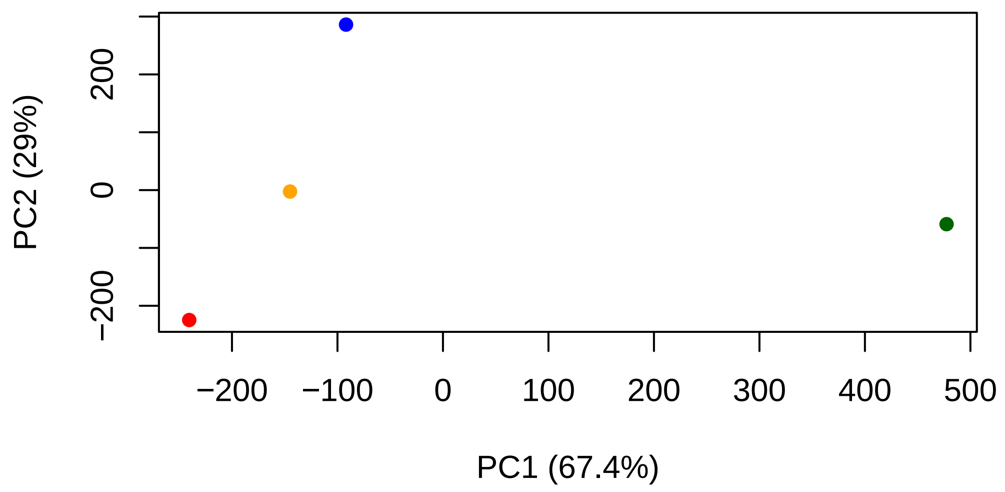
```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$class
[1] "prcomp"
```

We are after the `pca$x` result component to make our main PCA plot.

```
pca$x
```

```
                 PC1         PC2        PC3           PC4
England   -144.99315   -2.532999 105.768945 -9.152022e-15
Wales     -240.52915 -224.646925 -56.475555  5.560040e-13
Scotland   -91.86934  286.081786 -44.415495 -6.638419e-13
N.Ireland  477.39164  -58.901862  -4.877895  1.329771e-13
```

```r
mycols <- c("orange", "red", "blue", "darkgreen")
plot(pca$x[,1], pca$x[,2], col=mycols, pch=16, xlab = "PC1 (67.4%)", ylab = "PC2 (29%)")
```



Another important result from PCA is how the original variables (in this case foods) contribute to the PCs.

This is contained in the `pca$rotation` object - folks often call this the “loading” or “contributions” to the PCs

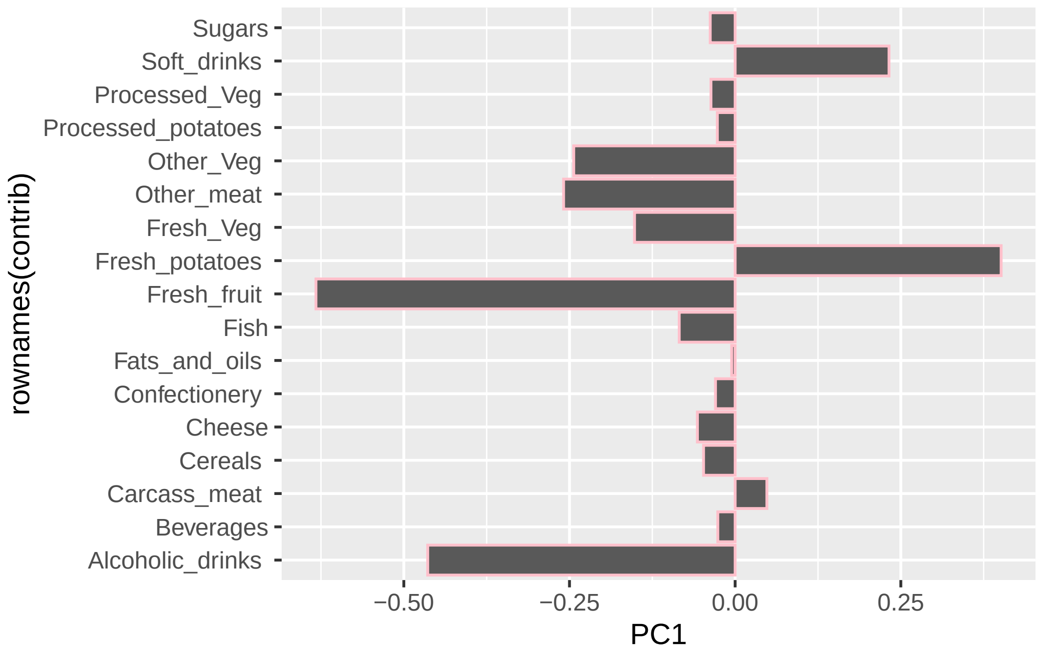
```r
pca$rotation
```

```
                        PC1          PC2         PC3          PC4
Cheese         -0.056955380  0.016012850  0.02394295 -0.409382587
Carcass_meat    0.047927628  0.013915823  0.06367111  0.729481922
Other_meat     -0.258916658 -0.015331138 -0.55384854  0.331001134
Fish           -0.084414983 -0.050754947  0.03906481  0.022375878
Fats_and_oils  -0.005193623 -0.095388656 -0.12522257  0.034512161
Sugars         -0.037620983 -0.043021699 -0.03605745  0.024943337
Fresh_potatoes  0.401402060 -0.715017078 -0.20668248  0.021396007
Fresh_Veg      -0.151849942 -0.144900268  0.21382237  0.001606882
Other_Veg      -0.243593729 -0.225450923 -0.05332841  0.031153231
```

```
Processed_potatoes  -0.026886233  0.042850761 -0.07364902 -0.017379680
Processed_Veg       -0.036488269 -0.045451802  0.05289191  0.021250980
Fresh_fruit         -0.632640898 -0.177740743  0.40012865  0.227657348
Cereals             -0.047702858 -0.212599678 -0.35884921  0.100043319
Beverages           -0.026187756 -0.030560542 -0.04135860 -0.018382072
Soft_drinks          0.232244140  0.555124311 -0.16942648  0.222319484
Alcoholic_drinks    -0.463968168  0.113536523 -0.49858320 -0.273126013
Confectionery       -0.029650201  0.005949921 -0.05232164  0.001890737
```

We can make a plot along PC1.

```r
library(ggplot2)

contrib <- as.data.frame(pca$rotation)

ggplot(contrib) +
  aes(PC1, rownames(contrib)) +
  geom_col(col = "pink")
```


Sugars
Soft_drinks
Processed_Veg
Processed_potatoes
Other_Veg
Other_meat
Fresh_Veg
Fresh_potatoes
Fresh_fruit
Fish
Fats_and_oils
Confectionery
Cheese
Cereals
Carcass_meat
Beverages
Alcoholic_drinks
rownames(contrib)
PC1